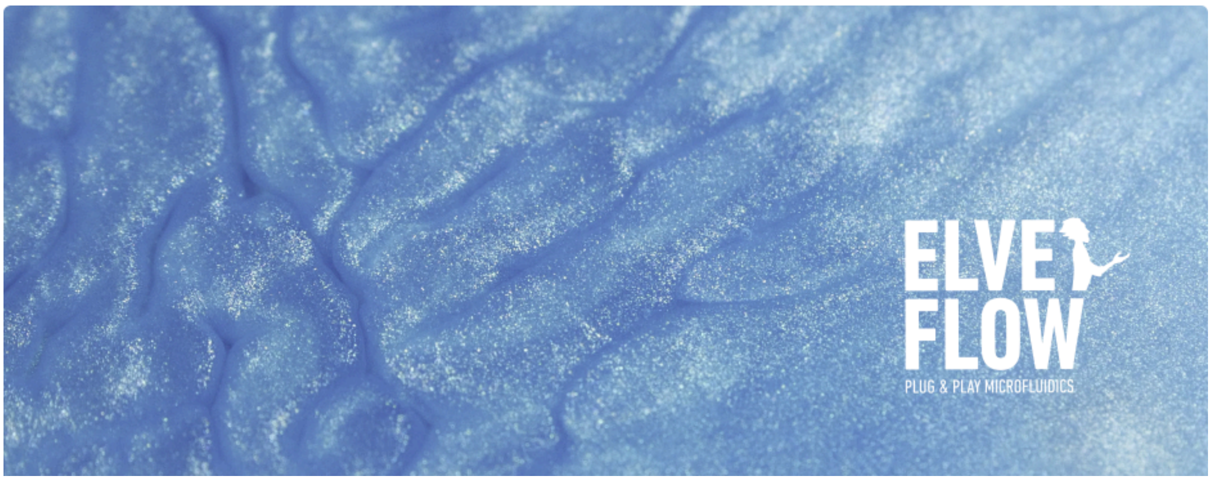
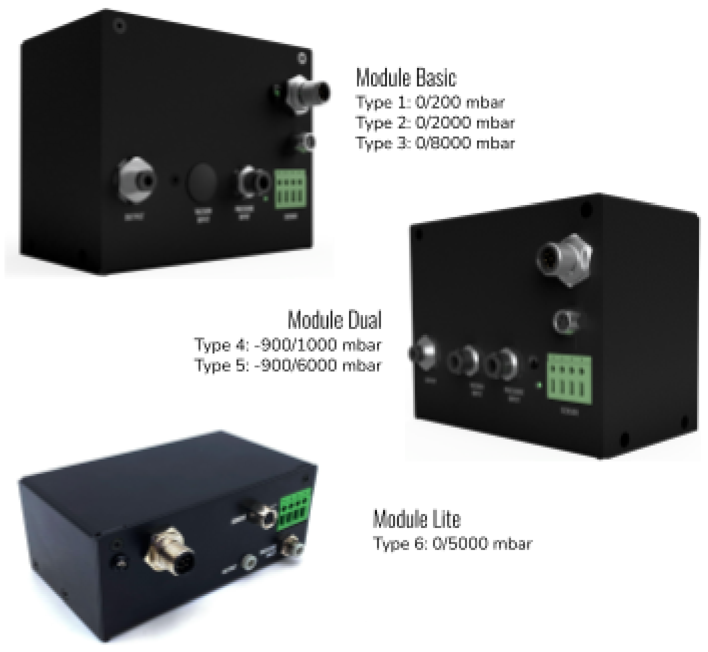


ADVANCED PRESSURE CONTROLLER COMMUNICATION PROTOCOL



This document provides the information needed to communicate with the Advanced Pressure Controller module through direct UART communication.

UART Communication Protocol for Advanced Pressure Controller / Version: 01.01.01 / Date: August 2024

Introduction

The Advanced Pressure Controller can be controlled directly via an adapter (intermediary between the module and the computer) or it can be controlled via an Advanced Control Center. If you are using the Pressure Controller with the Advanced Control Center, refer to the Control Center communication protocol documentation for additional information.

When using the adapter, we highly recommend connecting the M12 cable between the adapter and the device before connecting the USB cable and the DC power supply. Also, please remove USB cable and DC power supply from the adapter before the M12 cable between the adapter and the Pressure Controller.

Its main functions are pressure control and reading sensor values. It can also, configured appropriately :

- perform a PI control in order to modulate the pressure output according to sensor values read by the module
- perform a remote PI control (when the pressure controller is paired with a control center and another sensing module : another pressure controller or a sensor hub)
- perform pressure waveforms (a pressure profile over time, generic or custom)
- personalize sensor linear coefficients (slope and offset)

For more details on PI control and the relationship between flow rate and pressure, please check the user guide of the MFS flow sensors.

Table of contents

Serial connection settings

Baud rate: 230400

Data bits: 8

Stop bit: 1

Parity: none

Termination character: ‘\n’

Syntax

Command syntax

char 0: ‘<’ to start the query

char 1 to 5: command name

char 6: ‘?’ to read, ‘!’ to write

then ‘:’ to start a value. Can iterate over many arguments

Error handling

In the answer to any command, the first information displayed after the read / write character is the error code ‘|xx|’. It is two characters that indicate the error code associated with the request. ‘00’ means no error. Refer to the following table to check the possible error codes related to the Pressure Controller module:

Error code	Meaning
00	**No error**
C0	**Channel error: wrong channel requested**
L0	**Locking error: you do not have writing access to this parameter**
I0	**Impossible command: this query can not be processed**
P0	**Pause error: this command can not be processed while pause is set to 1**
NS	**No sensor connected to this channel**
B0	**Argument value out of bound**

Quickstart

A Pressure Controller module contains two main parts: a regulator unit capable of controlling an output pressure and a sensing communication head to plug sensors.

All the commands listed in this section and additional ones are grouped together in the commands table.

Pressure control

The Pressure Controller internal regulator is a high-speed, high-quality pressure regulator. It uses an input pressure to regulate an output pressure according to the user's request. The **PRESS** command is essential for controlling and measuring pressure, and it exclusively uses mbar values for writing and reading pressure values.

When the module is powered up, the pressure target is initially set to 0.

Calibration is unnecessary due to the high-quality calibration procedure conducted during device production. However, if you notice significant discrepancies between the pressure command and the physical output, it is crucial to carefully check the input pressure and inspect the tubing for any leaks. If a difference persists, you can adjust the command in the software using a linear calibration done with your own high precision pressure sensor, and contact in the meantime customer support for assistance.

Sensing

A Pressure Controller sensing communication head features a hybrid module that allows for direct analog reading from analog sensors, and includes an I2C head for communication with digital sensors.

Sensor control & PI regulation

Elveflow digital sensors are automatically detected and read by the device. It is possible to apply offset, slope and quadratic coefficients to the values measured by the sensor using **SENCA** command to set these coefficients.

To use Elveflow analog sensors with the device, it is required to initially set the correct sensor type using the **SENSO** command, in accordance with the sensor type correspondence table. This table matches analog sensors with their corresponding integer types for use in the SENSO command.

The module facilitates pressure PI regulation. This regulation can either be feedback looped to the module's own sensor or configured to 'listen' to a sensor from another module (such as Sensor Hub or another Pressure Controller). For instructions on how to set up a remote feedback loop, refer to the Control Center documentation.

Determine P and I values for a microfluidic setup

1. The first step is to size your microfluidic setup. Check that the range of the sensor matches the pressure range of the pressure controller. In other words, when applying the maximum available pressure value of the controller, the sensor should reach the maximum sensor value of its range. If the maximum pressure value leads to overflow of the sensor or does not reach the target value for the PI control, adapt the microfluidic setup by increasing or decreasing the set up's microfluidic resistance by adding or removing microfluidic resistance tubing. For more information on microfluidic resistance and PI control, refer to the MFS user guide on the Elveflow website.
2. Once you have a satisfying max pressure - max measurable flow rate correspondence, you can start by using I=0 and trying to find the best P possible (ie P value leading to best control).
3. Once you find a satisfying value, keep it and now work on the I value the same way.
4. You can refine P with the fixed I value from the previous step.
5. Iterate between step 3 and 4 until you're satisfied.

Once you find P and I parameters, they should be valid for every target, as long as you keep the same microfluidic setup.

Here are behaviors to observe and use for the P and I refining :

- decreasing the P gain reduces the overshoot
- decreasing the P gain reduces oscillations
- decreasing the I gain reduces the overshoot
- decreasing the I gain reduces oscillations

Implementation of a PI control

As a general guide, here's how to perform PI control, for instance, to achieve 500 µL/min:

1. Set pressure limits (applied when PI control is active) to prevent sensor saturation if desired, by using the UART command **USRPL**. It's important to note that the upper pressure limit should still permit the system to reach the sensor's maximum value.

	Send	**Receive**
Syntax	<:USRPL!:0:750	<USRPL!|00|00000.00:00750.00

Explanation	0 = pressure limit inf 750 = pressure limit sup	00000.00 = pressure limit inf 00750.00 = pressure limit sup

2. Set the desired pressure/flow rate target.

	Send	**Receive**
Syntax	<SENSC!:500	>SENSC!|00|00500.00
Explanation	Arguments : **float: target value**	

3. Set up the P and I values and launch PID control

	Send	**Receive**
Syntax	<SETPI!:0:**0.15**:**0.23**	>SETPI!|00|00:**00000.15**:**00000.23**
Explanation	Index 0 because there is only one channel on Pressure Controller Arguments : **float: P value to set**	

	float: I value to set	

4. Launch PI control using **PIRUN**

Remote PI regulation

You can also, before the procedure described above, 'connect' a Pressure Controller to another Pressure Controller or a Sensor Hub in order to **adjust the pressure on the first Pressure Controller while 'listening' to the sensor values measured by the second Pressure Controller or Sensor Hub**. This feedback loop and PI control is then deported.

Please refer to the Advanced Control Center communication protocol for more information on this feature, as you need the Pressure Controller and the secondary sensing device (another Pressure Controller or a Sensor Hub) both connected to a common Control Center in order to set up such a remote PI control.

Custom waveforms

Four custom waveforms (indexed from 1 to 4) are available for pressure or sensor control (i.e., control based on internal pressure sensor values of the regulator or external sensor values).

These custom waveforms are expressed through 6000 value points (from index 0 to 5999) separated in time by 10 ms. This implies that each custom waveform represents a 60-second period, which is set to repeat continuously.

These custom waveforms are saved in the device's hard memory (EEPROM) and loaded at startup. If you modify a waveform and wish to use it, the device must be restarted.

How to modify a custom waveform

1) Use the **WAVCI!** command to modify the value of one index of a specific custom waveform. Repeat this for as many value indices as necessary.
   - Example: <WAVCI!:1:149:20 sets the 149th value (out of 6000) of the 1st custom waveform to the value 20

2) Use the **WAVCE!** command to manually launch the saving of the custom waveform in the device's hard memory (EEPROM).

- Example: <WAVCE!:1. This command writes custom waveform 1 in the device's hard memory (EEPROM), ensuring the modification is saved.

3) Restart the device and verify that the correct value has been written by using the **WAVCI?** command.

   - Example: <WAVCI?:1:149> - This reads the 149th value (out of 6000) of the 1st custom waveform.

How to select a custom waveform to be used for control

Use the **WAVCT!** command to set the custom waveform that you want to use for control.

- Example: <WAVCT!:1:150 custom waveform indexed 1 will be used for control with an offset of 150 points

- Example: <WAVCT!:0:0 sets control to static amplitude control (index 0 (first argument) corresponds to default amplitude control)

How to check what custom waveform is in use

Use **WAVCT?** to get the custom waveform index currently used

- Example: <WAVCT? and answer : >WAVCT?|00|2:0000:0020.000 means that custom waveform used is index 2 with an offset of 0 and current pressure/sensor query is 20

List of commands

- W means 'write', ie set value, available for this command if ticked
- R means 'read', ie get value, available for this command if ticked
- 'Mandatory arguments' corresponds to mandatory arguments to attach to command in R mode
- 'Arguments' corresponds to additional mandatory arguments to attach to command in W mode after the 'Mandatory arguments'
- 'Arguments' also corresponds to additional values in the answer resulting from the sent command (in W or R mode), after the 'Mandatory arguments'

Other auxiliary tables after this one.

Parameter	Mandatory arguments	Arguments (argument type: argument name)	W	R	Number of characters returned	Example query	Typical answer	Note
PINGA		**float:** pressure value of regulator **float:** sensor value **int:** sensor type		X	35	<PINGA?	>PINGA?|00|00325.12:00124.13:04:00	

		bool : injecting						
IDN		**str:** device name		X	22	<_IDN_?	>_IDN_?|00|PRESSCONTR	
PRESS		**float:** pressure target in mbar	X	X	20	<PRESS? <PRESS!:364	>PRESS?|00|00498.98 >PRESS!|00|00364.00	
SENSC		**float:** sensor target value	X	X	20	<SENSC? <SENSC!:500	>SENSC?|00|00500.00 >SENSC!|00|00500.00	**Set or ask sensor value target and start PID**
WAVET		**int:** waveform type **float:** max value **float:** min value	X	X	50	<WAVET? <WAVET!:500:200:100:0	>WAVET?|00|01:00500.00:00200.00:00100.00:00000.00 >WAVET!|00|01:00500.00:00200.00:00100.00:00000.00	**Set the specified classic waveform to be used for PID control** **Waveform types:**

		float: period in s **int:** phase in degree						**00:** basic amplitude control **01:** sine **02:** square **03:** triangle **04:** linear
PIRUN		**bool :** PID runs (switch between pressure control and sensor control) **bool :** PID pause	**X**	**X**	17	<PIRUN? <PIRUN!:1:0	>PIRUN?|00|00:00 >PIRUN!|00|01:00	**Get or set PI control (0 for pressure (regulator) or 1 for sensor (external))** **and pause status (pressure output will freeze to last set pressure value when paused (pause status activated = 01), control will resume where it was left of when play)** **/!\ changing PI control resets PI parameters (target & accumulation error)**

DEVSN		**str:** SN		X	27	<DEVSN?	>DEVSN?|00|B00004	
FIRMV		**str:** firmware version		X	21	<FIRMV?	>FIRMV?|00|v01.03.01	
RESET						<RESET		**reset firmware, i.e. soft reset of the device, i.e. simulates a power off-power on which resets all volatile variable (not saved in hard memory)**
SENSO	**int** : channel (only 1)	**int:** sensor type	X	X	17	<SENSO?:1 <SENSO!:0:21	>SENSO?|00|01:04 >SENSO!|00|01:21	**Sensor types:** **00:** no sensor **01:** MFSD1 **02:** MFSD2 **03:** MFSD3

								04: MFSD4 **05:** MFSD5 **21 to 44:** analog sensors **(W mode only compatible with analog sensors)**
SENCA	**int :** channel (only 1)	**float:** sensor slope **float:** sensor offset	X	X	32	<SENCA?:2 <SENCA!:2:2.3 1:0.04	>SENCA?|00|02:00001.00:00000.00 >SENCA!|00|02:00002.31:00000.04	
SENRA	**int :** channel (only 1)	**int:** sensor measured rate		X	17	<SENRA?:3	>SENRA?|00|03:119	
SENSI	**int :** channel (only 1)	**int:** start or stop status **float:** volume injected since start	X	X	26	<SENSI!:1:1	>SEINT!|00|01:01:00000.00	**to start the volume injection : 1**

		(in uL)						**to stop the volume injection: 0** **when stopping the injection, the final volume value persists until another injection is started again**
SEINT	**int :** channel (only 1)	**int:** start or stop instruction **float:** sensor value integral over time since start	X	X	26	<SEINT?:1 <SEINT!:1:0	>SEINT?|00|01:01:00010.34 >SEINT!|00|01:00:00011.26	**to start the integration : 1** **to stop the integration : 0** **when stopping the integration, the final integral value persists until another integration is started again**
SENRE	**int** : channel (only 1)	**int:** sensor resolution	X	X	17	<SENRE?:1 <SENRE!:1:8	>SENRE?|00|01:04 >SENRE!|00|01:8	**Sensor resolution corresponds to the number of bits used to code the sensor value (see Acquisition time table for sensor resolution for details)** **Compatible only with digital sensors type 1 to 5**

								/!\ wait 500ms before sending another UART command (necessary time to let the sensor parameter update)
SENLT	**int :** channel (only 1)	**int:** liquid type	X	X	17	<SENLT?:0 <SENLT!:0:3	>SENLT?|00|00:00 >SENLT!|00|00:03	**compatible with sensors MFSD2, MFSD3 and MFSD4 only** **see Correspondence table for liquid types** **caution : liquid type is reset at each power up**
REGSN		**str:** regulator SN		X	22	<REGSN?	>REGSN?|00|XXXXXXXX	
SETPI		**float:** P	X	X	29	<SETPI?	>SETPI?|00|00010.00:00003.00	

		float: I				<SETPI!:11:2.2	>SETPI!|00|00011.00:00002.20	
ERLOG		**float :** PI error **bool :** physical error marker	X	X	27	<ERLOG? <ERLOG!:0	>ERLOG?|00|000002345.32:00 >ERLOG!|00|000000000.00:00	**Get PID error or set PID error to custom value** **When physical error marker is true (= 1), it means the PI control accumulated error is drifting (if 1, automatically pauses PI control and deco/reco sensor)**
USRPL		**float** : pressure min allowed during PI sensor control **float** : pressure max allowed during PI sensor control	x	x	29	<USRPL? —------------------------ <USRPL!:500:1200	>USRPL?|00|00500.00:01200.00 —--------------------- >USRPL!|00|00500.00:01200.00	
WAVCI	**int:** waveform index (1 to 4)	**float:** waveform value	X	X	28	<WAVCI?:1:149 <WAVCI!:1:149:	>WAVCI?|00|01:0149:0020.000	**get or set the value of the index (from 0 to 5999) of a specified custom**

	int: waveform value index (0 to 5999)					20	>WAVCI!|00|01:0149:0020.000	**waveform (identified by its index)**
WAVCE	**int:** waveform index (1 to 4)		**X**	**X**	14	<WAVCE?:1 <WAVCE!:1	>WAVCE?|00|01 >WAVCE!|00|01	**read or write from/to the device's EEPROM the specified custom waveform (identified by its index)**
WAVCZ	**int:** waveform index (1 to 4)		**X**		14	<WAVCZ!:1	>WAVCZ!|00|01	**reset specified custom waveform (identified by its index) values to 0**
WAVCT		**int:** custom waveform index **int:** offset point to start custom waveform from	**X**	**X**	19	<WAVCT? <WAVT!:01:100	>WAVCT?|00|01:0100 >WAVCT!|00|01:0100	**Set the specified custom waveform to be used for PID control and immediately starts running the custom waveform**

Correspondence table for sensor types

Type id	Unit in output	Sensor type
0		No sensor connected
1	µL/min	MFS1 digital
2	µL/min	MFS2 digital
3	µL/min	MFS3 digital
4	µL/min	MFS4 digital
5	µL/min	MFS5 digital
6-20		Reserved
21	µL/min	MFS1 Analog
22	µL/min	MFS2 Analog
23		Reserved
24	µL/min	MFS3 Analog
25	µL/min	MFS4 Analog
26	µL/min	MFS5 Analog
27-29		Reserved
30	mbar	MPS0 Analog
31	mbar	MPS1 Analog
32	mbar	MPS2 Analog
33	mbar	MPS3 Analog
34	mbar	MPS4 Analog
35	mbar	MFP

36-39		**Reserved**
40	**mV**	**Bubble detector**
44	**mV**	**Custom**

Acquisition time table for sensor resolution

Resolution mode	Resolution (bit)	Processing Time Min. (ms)	Processing Time Typ. (ms)	Processing Time Max. (ms)
1	**9**	**0.5**	**0.8**	**0.9**
2	**10**	**1.0**	**1.3**	**1.5**
3	**11**	**2.0**	**2.4**	**2.6**
4	**12**	**4.1**	**4.6**	**4.9**
5	**13**	**8.2**	**8.9**	**9.4**
6	**14**	**16.4**	**17.5**	**18.5**
7	**15**	**32.8**	**34.8**	**36.7**
8	**16**	**65.5**	**69.3**	**73.2**

Correspondence table for liquid types

This table applies to the following sensors: digital sensors MFS2, MFS3 and MFS4.

Liquid ID	Corresponding liquid
0	Water
1	IPA
2	Not applicable (liquid type doesn’t mean anything for the connected sensor)